PROCEEDINGS

55. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

13 - 17 September 2010

Crossing Borders within the ABC

Automation,

Biomedical Engineering and

Computer Science

Faculty of
Computer Science and Automation

www.tu-ilmenau.de

TECHNISCHE UNIVERSITÄT
ILMENAU

Impressum
Published by

Publisher:	Rector of the Ilmenau University of Technology Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c. Peter Scharff
Editor:	Marketing Department (Phone: +49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de) Faculty of Computer Science and Automation (Phone: +49 3677 69-2860) Univ.-Prof. Dr.-Ing. habil. Jens Haueisen
Editorial Deadline:	20. August 2010
Implementation:	Ilmenau University of Technology Felix Böckelmann Philipp Schmidt

USB-Flash-Version.

Publishing House:	Verlag ISLE, Betriebsstätte des ISLE e.V. Werner-von-Siemens-Str. 16 98693 Ilmenau
Production:	CDA Datenträger Albrechts GmbH, 98529 Suhl/Albrechts
Order trough:	Marketing Department (+49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de)

ISBN: 978-3-938843-53-6 (USB-Flash Version)

Online-Version:

Publisher:	Universitätsbibliothek Ilmenau ilmedia Postfach 10 05 65 98684 Ilmenau

Home / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

FIRST THOUGHTS ON A SECURE AND RELIABLE PEER-TO-PEER SERVICE PLATFORM

Muhammad Ikram, Kamill Panitzek, Max Mühlhäuser, Thorsten Strufe

TU Darmstadt, Germany

ABSTRACT

Harnessing the P2P paradigm to provide a distributed service platform for novel application domains requires entirely new management schemes and raises new security challenges. Critical resources actively have to be allocated and shared in a decentralized manner. Resilience to adverse behavior has to be achieved in the open, cooperative environment. We aim at analyzing the requirements and providing some general building blocks for the provision of such a general service platform, focussing on the challenges to provide fair and reliable load balancing while keeping the services available and the platform secure. This paper represents work in progress and outlines our current research perspectives.

***Index Terms*—** distributed computing, distributed control, availability

1. INTRODUCTION

The peer-to-peer (P2P) paradigm [18] has received enormous attention, triggering a movement to distribute traditional client-server-based systems further to decentralization and self-organization. Initially, P2P systems were designed to efficiently distribute content over the Internet, with the intention to decrease centralized control. The majority of developed systems hence mainly provided means to build applications in domains aiming at content dissemination. These mainly include file sharing, voice over IP, instant messaging, and video streaming. P2P systems in these cases successfully distribute and balance the load to provide services automatically, thus achieving enhanced performance (and liberty).

Extending this potential to the domain of distributing services demands additional functionality. Distributed service execution, access, and control systematically differs from content distribution. In contrast to content, the subject of distribution in this case represents running applications including their system state (cmp. Fig. 1). A P2P-based service platform consequently has to provide means for differing types of access (remote invocation vs. retrieval of code and local execution). It additionally needs to facilitate proactive load balancing of services, which has to account for the consistency of distributed services, since service access, unlike in the content distribution scenario, does not inherently distribute the load.

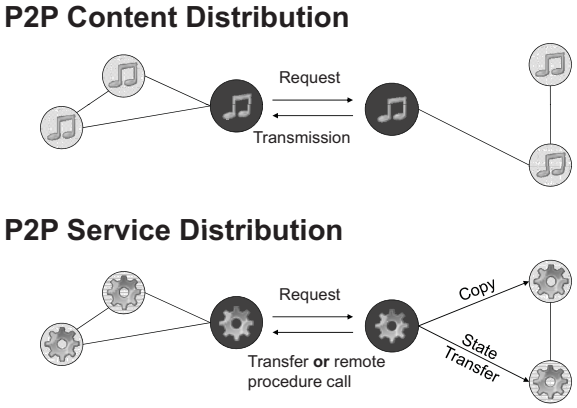

Fig. 1. P2P Content Distribution vs. P2P Service Management

Service platforms, frequently subsumed under the label of "cloud computing", are generally implemented in a centralized fashion, today. Some first approaches exist that allow for service discovery using P2P mechanisms. Even though a good basis for extension, they do not leverage the distributed resources of users for computational purposes. Other approaches to realize distributed computing have been designed under the label of "Grid Computing" like P2P-Grid for instance [22], which, however, come with an immense management overhead. Open Cloud Computing platforms, like, e.g., the freely available system Eucalyptus [17] could be used as first steps to realize a service platform. These systems depend on homogeneous environments though, and hence are better taylored towards service distribution inside organisational domains, rather than the open context of P2P systems. They additionally are characterized by implementing a centralized control unit to coordinate the computing nodes and assign jobs, which would have to be distributed to cater for an open system following the P2P paradigm. Summarizing, we acknowledge a large body of very good related work, but still see the need to further the field, by introducing the

This work in parts has been supported by the DFG Research Group 733 (QuaP2P) and by the IT R&D program of South Korea's MKE/KEIT under grant number 10035587.

concept of P2P-based service platforms.

Potential application domains for such a service platform are abound. Considering their character, which define the requirements to the service platform, we identify five examples, each representing extreme demands to some class of requirements. They include emergency first response, online social networks (OSN), massively multi-player online gaming (MMOG), social-IPTV, and distributed computing. The functional, operational, and security requirements of these scenarios greatly differ.

In this paper we propose a unified, generic, and secure service platform for P2P-based application scenarios. The contributions of this paper include:

- We describe P2P service scenarios to reveal their functional and operational requirements, thus identifying the challenges to the design of a general service platform.
- We identify the core functional modules needed in this platform and propose a first architectural design.
- Further, we also anlayze security threats and vulnerabilities in these application scenarios, with a special focus on their implementation in an open, decentralized manner. We sketch an initial overview of these threats and the basic security services needed in such a service platform. designing a service platform.

The rest of the paper is organized as follows: Section 2 discusses functionality and challenges of our targeted application scenarios. Section 3 identifies requirements and Section 4 presents our design of a secure service platform. We present adversary models, security requirements, and threats in Section 5 and conclude the paper in Section 6.

2. APPLICATION SCENARIOS

As mentioned above, there are applications in numerous scenarios that could leverage on harnessing the P2P paradigm. In the following subsections, we describe a selected application scenarios including core challenges they pose, which need to be addressed in a secure service platform.

2.1. Emergency First Response

The users in emergency first response scenarios (EFRS), first responders, communicate using resource restricted, mobile devices, such as PDAs, mobile phones, or laptops to share information in disasters. Such communication scenarios normally have only limited, or even no infrastructures. The scenario is characterized by a dominance of insecure and unreliable wireless links, and a rapidly changing environment. Moreover, multiple, existing wireless networks (wireless infrastructure networks, ad hoc networks, and wireless sensor networks) vastly differ in terms of communication mechanisms, packet types, and resources. An emergency first response application hence has to deal with a wide heterogeneity of devices and communication networks, and with highly dynamic environments. Any application used to communicate in such a scenario hence needs to be *reliable*, *secure* and *resilient*.

2.2. Online Social Networks

Social networking applications are among the most popular services on the Internet, today [3][4]. Users of these Online Social Networks (OSN) create profiles, share information, and communicate in pairs or groups. Users can befriend each other to share content and information across the network (like, e.g., in facebook[1]), communicate via streams of published messages others subscribe to (like, e.g., in twitter [2]; we refer to previous work in which we have given a broader overview of the domain of online social networks [8]).

Users of OSN tend to change and up-date their profile regularly to present achievements in real live, as well as their mood, feelings, thoughts on important topics, etc. This information needs to be consistent throughout the whole network to avoid access to out-dated profiles and hence sustain a satisfying user experience. This is a huge challenge especially in the context of P2P systems, since information may be stored on different nodes and therefore needs to be changed uniformly to achieve *consistency*. Profiles in OSN are requested by many users and need to be accessible anytime, to avoid losing participants of the OSN. Therefore the system has to guarantee a certain level of *availability*. *Privacy* is yet another predominant requirement in OSN. Providers of OSN have introduced rudimentary mechanisms to give users the feeling of being able to control acces to their data. The information ultimately is stored at the premises of the providers, though, who hence are capable of using it for whichever purpose they please.

2.3. Massively Multi-player Online Games

Massively multi-player online games have a huge online market share, and require large amounts of networking, processing, and storage resources. Player of these games can choose an avatar that can gain experience by harvesting online money, credits, or points, while playing and communicating with other avatars in the virtual world.

Performance in general is a big issue in this scenario. Systems that manage MMOG and provide access to them need to promptly react with very low latency to user input, to guarantee a satisfying gaming experience. Interactions with the virtual world and other

[1] *Facebook:* http://www.facebook.com/
[2] *twitter:* http://www.twitter.com/

players need to be *consistent* through out the whole system. This means that if a user picks up an item, for instance, this item should no longer be available for other players in the game, and hence the information of picking up an item needs to be populated throughout the whole network.

Cheating is a common problem in MMOG. Users keep trying to achieve an advantage over others by means of interacting with the system in an inappropriate way, thus trying to trick the system to maximize their health, wealth, or reputation. Detection of cheating is comparibly easy in centralized, yet considerably more complex in distributed game management. Therefore, functionality is needed to prevent users from single or distributed cheating [25] to provide a good gaming experience to all users.

2.4. Social IPTV

Social IPTV provides mechanisms to share multimedia content over a P2P streaming system. Shared video content in this scenario is combined with immersible text, gesture, or/and video, audio, or text responses from users to share opinions and comments on the ongoing video program, to combine entertainment with social interaction and increase user experience.

Such systems require best *performance* to provide high definition videos. They hence are characterized by a strong demand for high *throughput* in the network to achieve acceptable presentation quality of the video content. The huge variety of devices in this application scenario demand for openness of the system: the streaming platform should be able to handle the *heterogeneity* of connected devices, which may span the whole range from mobile phones, over TV sets, to large video walls. TV is consumed by many users with different knowledge regarding computers and technology, ranging from experts to laypersons. They interact with such systems in different ways so the system needs to be *flexible* enough to please the needs of all user types. Furthermore, commercial deployment as well as the active and open communication in social IPTV causes serious *security* and *privacy* issues. Attacks ranging from DoS [11], over content theft, spam, eavesdropping, to spying on users have to be expected and handled.

2.5. Distributed Computing

Distributed computing (DC) and cloud computing [23] have emerged to address an explosive growth of web-connected devices, and handle massive amounts of data. It is defined and characterized by massive scalability and new Internet-driven services, innovative applications, and currently spurs entirely new business models in online economics. Enterprises are attracted to cloud computing due to potential savings in IT and management expenditures. Considering the P2P paradigm, users are enabled to share memory and processing cycles in a community and to increase their own computational performance leveragin on other users' resources on demand.

The range of difference in applications that may potential be deployed in a DC scenario demands high *flexibility* of the system. Additionally, the consumers' data is processed in "the cloud", i.e. on machines they don't own nor control. The threats of *theft*, *misuse* or *unauthorized access* are a direct consequence from this characteristic. DC hence requires secure data handling to prevent illegitmate access to sensitive information.

3. TOWARDS A PEER-TO-PEER SERVICE PLATFORM

In the last section we have presented various application scenarios that can be implemented using distributed techniques. Every scenario consists of different applications and for every application one special system can be built to fit the requirements of the given scenario resulting in a huge variety of systems. Our main idea is to provide one integrated platform to easily distribute all these applications using P2P networks. Instead of building one system for every application our idea is to implement applications and distribute them like content in file sharing networks on top of a general service platform. Since an application can be composed of different types of services and also provide certain services a unified service platform could deliver basic development tools to implement these services and hence applications.

Comparing the management of distributed services with the traditional approach of content distribution we identified one main difference between those two approaches (Figure 1). In file sharing networks a file is basically requested from one peer and delivered by an other. By downloading a file it is replicated automatically since the file is then available on both peers. This also distributes the load automatically onto other peers. Improvements of replication schemes and load balancing techniques are possible and have been focus of research in the past [2] [12]. In the case of a P2P service platform, there are two possible ways to provide the service to the requesting peer each time a service is requested: the peer can either download the service and execute it on its own machine or it can invoke the service on the remote peer and just receive the result of the execution. In the latter scenario a peer providing a service could quickly be overloaded with requests which further means it can't provide the service to additional requests anymore. The service state additionally has to stay consistent in case of load balancing, code migration, churn, or peer failure. To prevent peer overload, the serving peer has to detect resource shortcomings and actively replicate its service onto another peer before failing. To realize this behavior special mechanisms need to be developed.

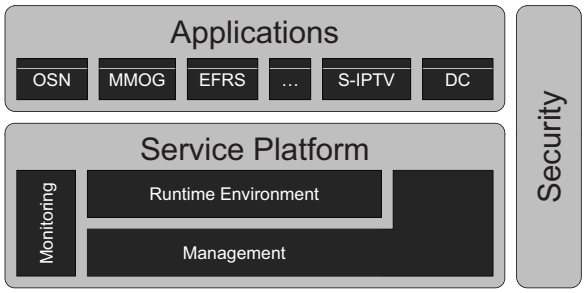

Fig. 2. P2P Service Platform Architecture

Four main groups of challenges to the distribution of services in a P2P-based service platform can be identified, especially in the context of the aforementioned application scenarios in Section 2:

Heterogeneous devices and infrastructures: the main characteristic of P2P systems are the very different environments on participating peers since the users connect with their laptops, desktop PCs or even mobile devices to the network. Our service platform needs to provide mechanisms to ensure homogeneous runtime environments on the participating peers.

Dynamic resources: since a service platform does not only share static content like files but also dynamic and running services it has to deal with consistency of execution state of these services.

Churn and resource limitations: P2P networks have to deal with churn, the arrival and departure or failure of peers. A P2P service platform has to provide mechanisms to counter the effects resulted by churn and to guarantee the availability of services. In cloud systems if the system is highly utilized providers can simply add new resources to the system and lower the utilization. In a P2P scenario this is not possible and hence the system has to equally distribute the load over the resources provided by the whole network to prevent single peers from overloading.

New attack methods: the distributed control needed to manage a service platform, as well as the migration of active mobile code in a P2P system, result in threats to the security that are new in the context of peer-to-peer systems. Therefore our platform has to provide methods to prevent attackers from misuse or sabotaging the system.

4. A FIRST ARCHITECTURAL SKETCH

We developed a basic component model of our P2P service platform illustrated in Figure 2. Our system design consists of three components: The **Application**, **Service Platform** and the **Security** component. Applications are built and run in the **Application** component on top of the service platform and depend on the respective scenario which we described in Section 2. The scenario inherits main requirements and challenges to the application which could be handled by different modules. Since some of the described requirements are in contrast to each other only important ones can be considered for one application scenario at a time. Only the required modules for a specific scenario would then be active.

The **Service Platform** component consists of three modules: *Runtime Environment*, *Management* and *Monitoring* module. These modules are characterized by a high interdependencies. The *Runtime Environment* modules represents the environment in which the services or applications are running in. This layer also can include virtualization techniques to guarantee a homogeneous runtime environment for the implemented services. The system has to find matching peers in the network with corresponding runtime environments for the executed service when replicating the service on to another peer or performing load balancing methods. The *Management* module is responsible for self organizing the complete system by initiating these replication and load balancing techniques through code migration, remote procedure calls or simply downloading services. The *Monitoring* module observes the whole system and provides other layers with information about the current system state as well as available resources on local and remote peers. Observation can be done by passive monitoring or active probing. Especially the Management layer gains a benefit from the Monitoring layer since actual system state is crucial to important control decisions like balancing the load or replicating services.

The third component describes the **Security** component of our approach. Since a P2P service platform has to be secure to a variety of threats from both outside as well as inside attacks the security component is an important part of our system design. We identified the security requirements and present the most crucial ones to our approach in the following section providing a basic adversary model.

5. SECURITY CONSIDERATIONS

In order to guarantee the system's availability we assume security to be an essential part of service platform. Adversaries may be outsiders or insiders. Outsiders on one hand are parties that are not participating in the system at all. Insiders on the other are nodes that belong to the P2P system, which do not follow the protocol as expected, or which exhibiting a behavior that is destructive for the overall system. Considering the fact that an insider will generally be in a better position for attacks and that the open character of the P2P service platform will allow effectively any party to participate, we subsequently focus on insider attacks.

In general, some adversaries may selfishly consume resources[3] without actively disrupting the system. Oth-

[3]*Network resources* - bandwidth, buffer space, connection descriptors, etc., *Application resources* - storage, processing, etc.,

ers may try to break down control [19], application [16] or both of the system. We assume that a malicious node is able to generate packets with arbitrary content [6] [7], including forged source IP addresses [9] [21] [20], and is able to examine packets xrouted through it. Attackers may be able to thwart performance of P2P systems by hindering communication among parties [19] [15]. A mixture of both these attacks might be possible where adversaries want to control the overall system and to resist threat-mitigation techniques [16]. We also assume that adversaries have knowledge of a system's protocols and can conspire with each other to reduce their cost, intensify the adversarial effects, and gather information by providing and disseminating forged information and feedback.

Because of their open infrastructure P2P-based systems are vulnerable to a wide range of security threats. Analyzing the scenario, we can identify generic security requirements for these P2P-based application scenarios, as shown in Figure 3. Due to the generality of the service platform, these are mainly in line with generally known security goals and given as follows:

Confidentiality - services have to prevent illegitimate access and to provide protection against eavesdropping or exposure of control and application specific information. Further services are needed to protect the privacy of credentials and IDs from identification, resource tracking [24], and linking[1].

Integrity - packets and credentials have to be protected against tampering and recipients of message or mobile-code to have to be enabled to verify that transmitted data was not altered or manipulated. Data freshness and reliable time synchronization [24] have to be provided for.

Availability - the platform has to ensure that the overall system and the hosted applications remain operational persistently, even in the presence of faulty and malicious nodes. In order to provide availability, the system has to be robust and resilient against denial of service (DoS) attacks.

Authentication - users need means to verifie that the communication parties, peers, are valid and legitimate.

Non-repudiation - the platform in some scenarios has to ensure that the communicating parties cannott repudiate nor refute the services they offered. In addition to end-to-end transport security, a proof of transaction or reception is required for commercial viability of P2P applications and services [13] [5].

The service platform has to provide security services to fulfill all above mentioned requirements. Applications from all mentioned scenarios though may only require a subset of them, depending on their characteristics and needs. The security services of a general P2P service platform (as given in Fig. 2) have to fulfill these security requirements.

Requirements		OSN	MMOG	EFRS	DC	S-IPTV
Confidentiality		+	0	+	0	+
	Privacy	+	+	+	+	+
Integrity		+	+	+	+	+
	Data Freshness	-	+	+	+	0
	Code Integrity	-	+	+	-	+
	Time Synchronization	0	+	0	+	-
Availability		-	+	+	+	+
	Robustness	0	+	+	+	0
	Resilience	-	+	+	+	0
Authentication		0	+	+	+	-
Non-Repudiation		+	0	+	+	0

Fig. 3. Security requirements for P2P-based application scenarios.

6. SUMMARY AND FUTURE WORK

This work in progress paper proposes a general purpose P2P service platform. The functionality, challenges, and requirements of various application scenarios and their security concerns for such a service platform are discussed. We introduced five different service scenarios to sketch possible application examples that may run inside such a service platform, with widely differing demands. Subsequently we derived the core requirements for a service platform that aims at supporting applications from all the scenarios described. Our own first sketch for such a service platform consists of a first architectural design, which includes various functional and security components. Analysing the environment we have identified general threats and security requirements for P2P based service platforms.

In our future work we are aiming at refining our design and developing the core services, both with respect to the pro-active load balancing as well as the protection of the platform and its hosted applications.

7. REFERENCES

[1] Joan Arnedo-Moreno and Jordi Herrera-Joancomarti. Maintaining unlinkability in group based p2p environments. In *WETICE*, 2009.

[2] Daniel Bauer, Paul Hurley, and Marcel Waldvogel. Replica Placement and Location using Distributed Hash Tables. In *Local Computer Networks*, 2007.

[3] Sonja Buchegger and Anwitaman Datta. A case for P2P infrastructure for social networks - opportunities and challenges. In *Wireless On-demand Network Systems and Services*, 2009.

[4] Sonja Buchegger et al. PeerSoN: P2P social networking - early experiences and insights. In *Workshop on Social Network Systems*, 2009.

[5] Michael Conrad. Non-repudiation mechanisms for peer-to-peer networks: enabling technology

for peer-to-peer economic markets. In *CoNEXT*, 2006.

[6] Cristiano Costa and Jussara Almeida. Reputation systems for fighting pollution in peer-to-peer file sharing systems. In *IEEE P2P*, 2007.

[7] Cristiano Costa et al. Fighting pollution dissemination in peer-to-peer networks. In *Symposium on Applied Computing*, 2007.

[8] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Safebook: A Privacy-Preserving Online Social Network Leveraging on Real-Life Trust. *IEEE Communications Magazine*, 2009.

[9] Carlton R. Davis, Jos Fernandez, and Stephen Neville. Optimising sybil attacks against p2p-based botnets. In *Malware'09*, 2009.

[10] John R. Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, 2002.

[11] Virgil D. Gligor. On denial-of-service in computer networks. In *International Conference on Data Engineering*, 1986.

[12] Jussi Kangasharju, KW Ross, and DA Turner. Optimizing file availability in peer-to-peer content distribution. In *INFOCOM*, 2007.

[13] Steve Kremer, Olivier Markowitch, and Jianying Zhou. An intensive survey of fair non-repudiation protocols. In *Computer Communications*, volume 25, 2002.

[14] Ruidong Li et al. A novel hybrid trust management framework for manets. *Distributed Computing Systems*, 2009.

[15] Petros Maniatis et al. Impeding attrition attacks in p2p systems. In *SIGOPS European workshop*, 2004.

[16] Seth James Nielson and Scott A. Crosby. A taxonomy of rational attacks. In *International Workshop on Peer-to-Peer Systems*, 2005.

[17] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus Open-Source Cloud-Computing System. In *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 124–131, Washington, DC, USA, 2009. IEEE Computer Society.

[18] Andy Oram. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Inc., 2001.

[19] Marius Portmann, Sebastien Ardon, and Aruna Seneviratne. Mitigating routing misbehaviour of rational nodes in chord. In *Symposium on Applications and the Internet-Workshops*, 2004.

[20] A. Singh et al. Eclipse attacks on overlay networks: Threats and defenses. In *INFOCOM*, 2006.

[21] Atul Singh et al. Defending against eclipse attacks on overlay networks. In *SIGOPS European workshop*, 2004.

[22] P. Uppuluri, N. Jabisetti, U. Joshi, and Y. Lee. P2P grid: service oriented framework for distributed resource management. *2005 IEEE International Conference on Services Computing*, 1:347–350, 2005.

[23] Luis M. Vaquero et al. A break in the clouds: towards a cloud definition. *SIGCOMM Computer Communication Review*, 2009.

[24] Xinyuan Wang, Shiping Chen, and Sushil Jajodia. Tracking anonymous peer-to-peer voip calls on the internet. In *CCS*, 2005.

[25] Jeff Yan and Brian Randell. A systematic classification of cheating in online games. In *NetGames*, 2005.